WHAT IS CLAIMED IS:

1   1.   A processor for offloading processing in a storage environment,

2 comprising:

3   a processor interface that interfaces said processor to a network processor

4 configured to perform a storage function; and

5   semaphore circuitry, coupled to said processor interface, that receives a signal

6 from said network processor, and that controls a semaphore related to said signal for locking and

7 unlocking access to data.

1   2.   The processor of claim 1, wherein said semaphore circuitry manages a

2 queue for access to said semaphore.

1   3.   The processor of claim 2, wherein said semaphore circuitry receives a

2 second signal from said network processor and removes a request from said queue in response to

3 said second signal when said network processor no longer desires said semaphore.

1   4.   The processor of claim 2, wherein said semaphore circuitry refrains from

2 sending to said network processor a second signal indicating said semaphore is unavailable,

3 whereby said network processor continues to wait for said semaphore and said semaphore

4 circuitry maintains ordered access to said queue.

1   5.   The processor of claim 1, wherein said signal comprises one of a plurality

2 of access requests for one of a plurality of semaphores, wherein said semaphore circuitry

3 manages said plurality of access requests in a plurality of queues, and wherein each of said

4 plurality of queues corresponds to a respective one of said plurality of semaphores.

1   6.   The processor of claim 1, wherein said semaphore circuitry comprises:

2   a command queue that stores said signal received from said network processor.

1   7.   The processor of claim 1, wherein said semaphore is a structure in a hash

2 array, and wherein said semaphore circuitry comprises:

40

3   a hash key generator that performs a hashing function on said signal for accessing

4   said hash array.

1   8.   The processor of claim 1, wherein said semaphore circuitry comprises:

2   an update engine that receives a second signal from said network processor

3   relating to a first process thread on said network processor, releases a lock on said semaphore

4   related to said second signal, and sends a third signal to said network processor associating said

5   semaphore with a second process thread on said network processor.

1   9.   The processor of claim 1, wherein said semaphore circuitry comprises:

2   a semaphore queue manager that manages a queue of a plurality of semaphores.

1   10.   The processor of claim 1, wherein said semaphore circuitry manages a

2   queue for access to said semaphore, wherein said semaphore circuitry comprises:

3   a hash key generator that performs a hashing function on said signal for accessing

4   a hash array, and that generates a hash key related to said signal, wherein said semaphore is a

5   structure in said hash array, and wherein said signal relates to a first process thread on said

6   network processor; and

7   an update engine, coupled to said hash key generator, that receives said hash key,

8   that releases a lock on said semaphore related to said hash key, and that sends a second signal to

9   said network processor associating said semaphore with a second process thread on said network

10   processor, wherein said first process thread and said second process thread are associated with

11   said semaphore in said queue.

1   11.   A method of controlling a processor for offloading processing in a storage

2   environment, comprising the steps of:

3   receiving a signal from a network processor configured to perform a storage

4   function;

5   controlling a semaphore related to said signal for locking and unlocking access to

6   data.

1   12.   The method of claim 11, wherein said step of controlling said semaphore

2   includes:

41

3   managing a queue for access to said semaphore.

1   13. The method of claim 12, further comprising:

2   receiving a second signal from said network processor; and

3   removing a request from said queue in response to said second signal when said

4 network processor no longer desires said semaphore.

1   14. The method of claim 12, further comprising:

2   refraining from sending to said network processor a second signal indicating said

3 semaphore is unavailable, whereby said network processor continues to wait for said semaphore

4 and said processor maintains ordered access to said queue.

1   15. The method of claim 11, wherein said signal comprises one of a plurality

2 of access requests for one of a plurality of semaphores, and wherein said step of controlling said

3 semaphore includes:

4   managing said plurality of access requests in a plurality of queues, wherein each

5 of said plurality of queues corresponds to a respective one of said plurality of semaphores.

1   16. The method of claim 11, wherein said step of controlling said semaphore

2 includes:

3   receiving a second signal from said network processor relating to a first process

4 thread on said network processor;

5   releasing a lock on said semaphore related to said second signal; and

6   sending a third signal to said network processor associating said semaphore with a

7 second process thread on said network processor.

1   17. The method of claim 11, wherein said signal comprises one of a plurality

2 of access requests for one of a plurality of semaphores, and wherein said step of controlling said

3 semaphore includes:

4   managing said plurality of access requests in a plurality of queues, wherein each

5 of said plurality of queues corresponds to a respective one of said plurality of semaphores;

6   receiving a second signal from said network processor relating to a first process

7 thread on said network processor;

8   releasing a lock on said semaphore related to said second signal; and

9   sending a third signal to said network processor associating said semaphore with a

10   second process thread on said network processor.

1   18.   A processor for offloading processing in a storage environment,

2   comprising:

3   a processor interface that interfaces said processor to a network processor

4   configured to perform a storage function; and

5   ordering circuitry, coupled to said processor interface, that tracks an incoming

6   order of incoming frames received by said network processor and controls an outgoing order of

7   outgoing frames transmitted by said network processor.

1   19.   The processor of claim 18, wherein said ordering circuitry controls said

2   outgoing order by comparing a stored frame number with an incoming frame number of an

3   incoming frame.

1   20.   The processor of claim 18, wherein said ordering circuitry receives a first

2   signal from said network processor, verifies an order of a frame associated with said first signal,

3   and sends a second signal to said network processor, wherein said second signal indicates one of

4   that said frame corresponds to an in-order frame and that said frame corresponds to an out-of-

5   order frame.

1   21.   The processor of claim 20, wherein when said frame corresponds to said

2   in-order frame, said ordering circuitry receives a third signal from said network processor after

3   said network processor has transmitted said frame, and said ordering circuitry tracks said

4   outgoing order in response to said third signal.

1   22.   The processor of claim 20, wherein when said frame corresponds to said

2   out-of-order frame, said ordering circuitry generates said second signal indicating that said frame

3   is to be stored by said network processor.

1   23.   The processor of claim 18, wherein said ordering circuitry receives a first

2   signal from said network processor, determines whether a particular frame stored by the network

43

3   processor is in order to be transmitted, and identifies said particular frame to said network

4   processor.

1   24.   The processor of claim 18, wherein said ordering circuitry comprises:

2   a command processor that processes a command from said network processor

3   relating to an incoming frame.

1   25.   The processor of claim 18, wherein said ordering circuitry comprises:

2   an update state machine that manages a plurality of queues corresponding to a

3   plurality of process threads of said network processor, wherein a queue of said plurality of

4   queues includes a frame structure corresponding to a frame received by said network processor.

1   26.   The processor of claim 18, wherein said ordering circuitry comprises:

2   a read buffer that stores data relating to a next frame for said network processor to

3   transmit.

1   27.   The processor of claim 18, wherein said ordering circuitry comprises:

2   a command processor that processes a command from said network processor

3   relating to an incoming frame;

4   an update state machine, coupled to said command processor, that manages a

5   plurality of queues corresponding to a plurality of process threads of said network processor,

6   wherein a queue of said plurality of queues includes a frame structure corresponding to a frame

7   received by said network processor; and

8   a read buffer, coupled to said update state machine, that stores data relating to a

9   next frame for said network processor to transmit, as identified by said update state machine.

1   28.   A method of controlling a processor for offloading processing in a storage

2   environment, comprising the steps of:

3   tracking an incoming order of incoming frames received by said network

4   processor; and

5   controlling an outgoing order of outgoing frames transmitted by said network

6   processor.

1   29.   The method of claim 28, wherein said step of controlling includes:

2   comparing a stored frame number with an incoming frame number of an incoming

3   frame.

1   30.   The method of claim 28, wherein said step of controlling includes:

2   receiving a first signal from said network processor;

3   verifying an order of a frame associated with said first signal; and

4   sending a second signal to said network processor, wherein said second signal

5   indicates one of that said frame corresponds to an in-order frame and that said frame corresponds

6   to an out-of-order frame.

1   31.   The method of claim 30, wherein when said frame corresponds to said in-

2   order frame, said step of controlling further includes:

3   receiving a third signal from said network processor after said network processor

4   has transmitted said frame; and

5   tracking said outgoing order in response to said third signal.

1   32.   The method of claim 30, wherein when said frame corresponds to said

2   out-of-order frame, said step of controlling further includes:

3   generating said second signal indicating that said frame is to be stored by said

4   network processor.

1   33.   The method of claim 28, wherein said step of controlling includes:

2   receiving a first signal from said network processor;

3   determining whether a particular frame stored by the network processor is in order

4   to be transmitted; and

5   identifying said particular frame to said network processor.

1   34.   The method of claim 28, wherein said step of controlling includes:

2   processing a command from said network processor relating to an incoming

3   frame;

4   managing a plurality of queues corresponding to a plurality of process threads of

5   said network processor, wherein a queue of said plurality of queues includes a frame structure

6   corresponding to a frame received by said network processor; and

7   storing data relating to a next frame for said network processor to transmit, as

8   identified by said step of managing.

1   35.   A processor for offloading processing in a storage environment,

2   comprising:

3   a processor interface that interfaces said processor to a network processor

4   configured to perform a storage function; and

5   timer circuitry, coupled to said processor interface, that receives a plurality of

6   signals, that manages a plurality of timers related to said plurality of signals, and that generates a

7   timing result when one of said plurality of timers is completed.

1   36.   The processor of claim 35, wherein each of said plurality of timers

2   corresponds to a respective element in a doubly-linked list of elements.

1   37.   The processor of claim 36, wherein said timer circuitry manages said

2   doubly-linked list by comparing a free-running timer value to a first value associated with a

3   current element of said doubly-linked list, changing a second value associated with said current

4   element when said first value matches said free-running timer value, associating said current

5   element with an end of said doubly linked list after said second value has been changed, and

6   moving on to a next element of said doubly-linked list.

1   38.   The processor of claim 36, wherein said timer circuitry comprises:

2   a cache configured to store at least a portion of said doubly-linked list, wherein

3   said portion includes a current element and a previous element.

1   39.   The processor of claim 35, wherein said plurality of timers correspond to a

2   plurality of respective elements in a plurality of doubly-linked lists of elements, and wherein

3   each of said plurality of doubly-linked lists corresponds to a respective time base period.

1   40.   The processor of claim 35, wherein said timer circuitry comprises:

2   an expired timer queue that stores a plurality of timing results corresponding to a

3   plurality of expired timers.

1   41.   The processor of claim 40, wherein said network processor periodically

2   accesses said expired timer queue via said processor interface.

1   42.   The processor of claim 40, wherein said timer circuitry sends an interrupt

2   to trigger said network processor to access said expired timer queue.

1   43.   The processor of claim 35, wherein said timer circuitry comprises:

2   an expired timer queue manager that controls filling said expired timer queue with

3   said plurality of timing results from a memory.

1   44.   The processor of claim 35, wherein said timer circuitry comprises:

2   a remover circuit that removes a timer from said plurality of timers in response to

3   a stop timer signal via said processor interface from said network processor.

1   45.   The processor of claim 35, wherein a first portion of said plurality of

2   timers are stored in an external memory, and wherein said timer circuitry comprises:

3   a cache configured to store a second portion of said plurality of timers; and

4   an arbiter circuit that controls memory access between said external memory and

5   said cache.

1   46.   The processor of claim 35, wherein said plurality of timers correspond to a

2   plurality of respective elements in a plurality of doubly-linked lists of elements, and wherein

3   each of said plurality of doubly-linked lists corresponds to a respective time base period, wherein

4   said processor further comprises:

5   an expired timer queue that stores a plurality of timing results corresponding to a

6   plurality of expired timers; and

7   an expired timer queue manager that controls filling said expired timer queue with

8   said plurality of timing results from a memory.

1   47.   A method of controlling a processor for offloading processing in a storage

2   environment, comprising the steps of:

3   receiving a signal from a network processor configured to perform a storage

4   function;

5   managing a timer related to said signal; and

6   generating a timing result when said timer is completed.

1   48.   The method of claim 47, wherein said step of managing said timer

2 comprises:

3   adding said timer to a doubly-linked list, wherein said doubly-linked list

4 comprises a plurality of timers.

1   49.   The method of claim 47, wherein said step of managing said timer

2 comprises:

3   adding said timer to one of a plurality of doubly-linked lists, wherein each of said

4 plurality of doubly-linked lists comprises a plurality of timers, and wherein each of said plurality

5 of doubly-linked lists corresponds to a respective time base period.

1   50.   The method of claim 47, further comprising:

2   filling an expired timing queue with said timing result.

1   51.   The method of claim 47, further comprising:

2   stopping said timer in response to a stop timer signal from said network processor.